NASA Contractor Report 182010
ICASE Report No. 90-21

ICASE

PERFORMANCE BOUNDS ON PARALLEL SELF-INITIATING DISCRETE-EVENT

David M. Nicol

Contract No. NAS1-18605
March 1990

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665-5225

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

90

Performance Bounds on Parallel Self-Initiating Discrete-Event Simulations*

David M. Nicol

Department of Computer Science

College of William and Mary †

March 1, 1990

Abstract

This paper considers the use of massively parallel architectures to execute discrete-event simulations of what we term "self-initiating" models. A logical process in a self-initiating model schedules its own state re-evaluation times, independently of any other logical process, and sends its new state to other logical processes following the re-evaluation. Our interest is in the effects of that communication on synchronization. We consider the performance of various synchronization protocols by deriving upper and lower bounds on optimal performance, upper bounds on Time Warp's performance, and lower bounds on the performance of a new conservative protocol. Our analysis of Time Warp includes the overhead costs of state-saving and rollback. The analysis points out sufficient conditions for the conservative protocol to outperform Time Warp. The analysis also quantifies the sensitivity of performance to message fan-out, lookahead ability, and the probability distributions underlying the simulation.

*A preliminary version of this paper appears in the Proceedings of the 1990 SIGPLAN PPoPP Symposium.

†Supported in part by the Virginia Center for Innovative Technology, by NASA grants NAG-1-060 and NAS1-18605, and NSF grant ASC 8819393

1 Introduction

The problem of parallelizing discrete-event simulations has recently received a great deal of attention. Parallel simulations are typically described as a collection of *Logical Processes*, or *LP*s. Each *LP* maintains its own simulation clock, and communicates with other *LP*s using time-stamped messages. We assume each *LP* executes on its own processor, as it might on a massively parallel architecture. The state of an *LP* at simulation time t depends on the contents of all messages that should be sent to it with time-stamps less than t. There are two primary ways in which an *LP* re-evaluates its state. One way is epitomized by a queueing network simulation: a job leaving one queue (*LP*) causes the receiving queue to re-evaluate its state. This is an example of a *message-initiating* model, because state re-evaluations at an *LP* are caused by messages sent from other *LP*s. A different method occurs when an *LP* alone determines when to re-evaluate its state. The *LP* will send messages to other *LP*s following a re-evaluation, because those *LP*'s eventual re-evaluations will require that information. However, the state messages do not cause the recipients to re-evaluate their state. The messages cause events that serve only to store the transmitted state information. This paper concerns such models; we will call them *self-initiating* models. As we later discuss, this class of models includes problems as diverse as the Ising spin simulation[14], and trace-driven multiprocessor cache simulations.

Synchronization has been a major concern of research in parallel simulation. One way of ensuring correctness is to block an *LP* from computing its state at t if there is any chance that it will later receive a message with time-stamp $s < t$. This type of blocking is an open invitation to deadlock: irregular and unpredictable synchronization requirements make parallelizing discrete-event simulations a non-trivial problem. Early research efforts focussed on developing deadlock-free synchronization protocols. Two schools of thought emerged. The *conservative* school studied protocols that maintain consistency in the simulation state: an *LP* is never allowed to advance its clock so far that it can receive a message in its past. *LP*s exploit specific information about the simulation model to avoid or break deadlock. The *optimistic* school proposed Time Warp, a scheme that permits an *LP* to advance its clock without blocking. When an *LP* does receive a message in its past, it "rolls back" its clock to the point of the temporal fault, and restores its state to one existing prior to the fault. Time Warp does not need to use specific model information. Indeed, a major attraction of Time Warp is its transparency to the simulation modeler.

Before parallel machines were commonly available, the debate between conservative and optimistic camps was largely philosophical. Then, as performance studies were published, no clear consistently best choice emerged. The earliest conservative protocols of Chandy and Misra were shown to suffer from serious performance problems on some queueing network simulations [22], but have recently been shown to work well on road network simulations [17]. Other conservative protocols, notably [13] and [21], achieved acceptable performance on some problems by exploiting information about the simulation model. Time Warp too was shown to achieve acceptable performance on some problems [3, 4]. The overhead costs of state-saving and rollback continue to be a major drawback to all optimistic schemes; hardware accelerators for these functions have been proposed [1, 5].

Throughout this debate, little analytic theory was developed to predict, explain, or bound the performance of parallel simulations. Exceptions are the detailed analyses developed in [9] and [18]. However, these studies are limited to two processors, and have not been extended. Theory for massively parallel simulations is now starting to appear. Wagner and Lazowska derive an upper bound on the speedups possible in a queueing network simulation [25]. Studies of Time Warp tend to assume negligible state-saving and rollback costs. Lin and Lazowska have shown that if Time Warp has no state-saving or rollback costs, and if "correct" computations are never rolled back, then Time Warp achieves optimality [11]. This is intuitive, because Time Warp aggressively searches for the simulation's critical path—if it is able to do so without cost, its performance must be optimal. Other analyses highlight the fact that Time Warp can "guess right" while conservative methods must block. Lipton and Mizell have shown that there is a certain asymmetry between optimistic and conservative methods: while it is possible for an optimistic method to arbitrarily outperform a conservative method, the converse is not true [12]. Madisetti, Walrand, and Messerschmitt [16] have developed a performance model that aspires to estimate the rate at which simulation time advances under an optimistic strategy such as Time Warp. They model the behavior of the system as a Markov chain, and include the cost of communication and of synchronization. Their analysis is exact for two processors, and approximate for a general number of processors. Their analysis is interesting in that it permits a study of different re-synchronization schemes. However, it does not address issues we attack directly, namely, bounds on optimal performance and sensitivity to message-fanout and lookahead ability.

Analytic studies of conservative protocols [15, 20] are of synchronous protocols—a significant departure from the field's roots in distributed systems. These studies have established the important property that performance of the studied methods scales up with increasing problem size and architecture. Furthermore, the analysis in [20] demonstrates that as the problem size increases relative to the architecture, performance under the method converges to optimality. The rate of convergence depends very much on the nature of the stochastic processes driving the simulation.

A number of issues have not yet been directly addressed analytically, and are the focus of this paper. Specifically, we place non-trivial upper bounds on optimal performance; we include the overhead costs of Time Warp in a model that bounds its performance from above; we study a new conservative protocol and place a lower bound on its performance; we give conditions under which the conservative protocol achieves better performance than Time Warp. In the course of these derivations we quantify (approximately) the sensitivity of performance to lookahead ability, message fan-out, and the probability distributions driving the simulation. All of these fore-mentioned factors are shown to have significant influence on performance;

performance improves as lookahead ability improves or as the variability of the probability distribution decreases, performance degrades as the message fanout increases. It is important to note that these conclusions are derived in the context of self-initiating simulation models only. Different but related results can be derived in the context of message-initiating models[1].

2 Model

We model a parallel simulation as a collection of N logical processors (LPs) named $LP_1, \ldots, LP_N$. Each logical processor has its own simulation clock. LPs communicate through the exchange of time-stamped messages. Viewed from the perspective of simulation time, an LP advances forward by executing some activity that we will call a *cycle*. In the self-initiating models we consider, at the end of one cycle the LP schedules the end of the next cycle, independently of any messages it may have received from other LPs. We let $C_i(j)$ denote the value of LP_i's clock at the end of the jth cycle. The length of simulation time that LP_i advances by executing its jth cycle is a random number X_{ij} from a distribution $\mathcal{F}$. Consequently, for every LP_i and cycle j

$$C_i(j) = \sum_{k=1}^{j} X_{ik}.$$

Assuming that the time increment variables are all independent, $C_i(j)$ can be interpreted as the time of the jth renewal in some renewal process [24] with inter-renewal distribution $\mathcal{F}$. We introduce communication to the model by assuming that each LP_i associates a set of K messages with the completion of each of its cycles. K is called the message *fanout*. Typically, these K messages are intended to inform "nearby" LP's of the new state just computed. The arrival of such a message at an LP may cause an event, but one that serves only to store the transmitted value. $K = 2$ might be appropriate in a 1D domain, $K = 4$ or $K = 8$ in a 2D domain, $K = 6$ or $K = 26$ would be appropriate in a 3D domain. It is important to note that under our formulation these message fanouts are part of the simulation model, and hence are independent of the synchronization protocol used. A message associated with the completion of LP_i's jth cycle has time-stamp $C_i(j)$.

The simulation is modeled as N statistically independent, concurrent renewal processes that communicate. Certain points in the analysis to follow are made possible by the assumption of statistical independence between the recipients of a common message. To support this need for independence we assume that the K recipients of a message are chosen uniformly at random from the set of all LPs, and that each LP independently chooses a new set of recipients each cycle. This assumption does not accurately model the behavior of any common simulation model, and is used purely to promote tractability. We have performed simula-

[1] *Performance Bounds on Parallel Message-Initiating Discrete-Event Simulations*, D. Nicol, in preparation

tion studies of our analytic model using "nearest-neighbor" communication, and have found that processor utilizations are only slightly higher than those achieved using the randomized communication patterns our analysis assumes. We may have some confidence therefore that the conclusions derived under the assumption of randomized communication are not completely off the mark.

We will consider two different forms for the probability distribution $\mathcal{F}$. In one form $\mathcal{F}$ has a continuous cumulative probability distribution function, implying that its associated renewal process is non-lattice[24][2]. This form excludes simulation models where time-increments move forward more discretely. We therefore also derive results under the assumption that $\mathcal{F}$ is a geometrically distributed random variable, with mean $1/p$, where $0 < p \leq 1$.

Depending on the simulation model, it may be possible to send the messages associated with the completion of a cycle before an LP actually executes that cycle. In some cases the content of the messages cannot be predicted, but the time of the messages can. In the former case we will say the simulation has *full-lookahead*, in the latter case we say it has *time-lookahead*. An example of a model with time-lookahead is the Ising spin simulation [14]. LPs model individual particles, each of which is "jiggled" by thermal effects, at random intervals. When a particle is jiggled its new magnetic spin is computed as a function of the spins of nearby atoms at that simulation time. The length of simulation time between jigglings defines a cycle. We are able to predict when next a particle will be jiggled—this time comes from a random number generator—but will not know the spins of nearby particles at that simulation time until the simulation actually advances that far.

An example of a model with full-lookahead (although it's a message initiating model) is a queueing network with a non-preemptive and load-independent queueing discipline. At the time a job enters service, say s, we can predict the time at which it will leave service, say t. In fact, we can notify the recipient queue of that job's arrival at time t. This is not to say that we can actually simulate up to time t. For example, one of the statistics we may be interested in is the average length of the queue at the time a job departs. To measure the queue length at t we need to receive any additional jobs that may arrive between times s and t. The lookahead ability derives from the fact that arrivals between s and t in no way affect the output behavior of the LP at time t.

Another example of a model with full-lookahead is a simple trace-driven multiprocessor cache simulation that estimates hit statistics, such as that described in [10]. An LP models one processor's cache; cycles are composed of the processing of a contiguous sequence of purely local memory references terminated by a reference to global memory, the "time" of any reference is the number of trace references preceding it[3]. An

[2]A non-negative random variable X is non-lattice if there does not exist any real number d such that $\sum_{n=0}^{\infty} \Pr\{X = nd\} = 1$

[3]note that this property would not be satisfied by a simulation that more accurately models the advancement of time, e.g., one that accounts more time for a miss than a hit. A weaker form of lookahead exists where the LP can put a lower bound on the time of its next global reference by assuming all local references will be hits.

LP sends messages to all other LPs whenever it makes a reference to global memory. By looking at its own trace the LP can predict the time and content of its future messages. Throughout this paper, protocols that exploit full-lookahead will do so by requiring an LP to send a message as soon as it is able to predict that message. We will still require that an LP not process a cycle with completion time t until all messages with time-stamps less than t have been received, because certain calculations internal to the LP (e.g. statistics gathering) may require that this monotonicity be preserved. Protocols that exploit time-lookahead do so by requiring an LP to send an *appointment* message containing the time of a future message as soon as it is able to predict that a message will later be sent with that time-stamp. Our analysis will be of simulations with full-lookahead. We will later remark on how that analysis can be extended to simulations with only time-lookahead.

We assume that processing a cycle requires one *tick* of real time. This permits us to view the progress of the simulation synchronously. While an LP will read all messages sent to it, at each tick, it need not process a cycle every tick; in fact, synchronization constraints may prevent it from doing so.

Some synchronization protocol must be used to ensure correctness. A conservative protocol prevents an LP from advancing so far that it can receive a message with a time-stamp smaller than its clock value. For example, imagine a situation where LP_i's clock is s and it will increment its clock to value t on the next cycle it processes. Imagine that LP_k will send a message to LP_i with time-stamp v, $s < v \leq t$, at the end of the $k+4th$ tick. A conservative protocol will ensure that LP_i is idle during ticks $k+1$ through $k+4$. An optimistic protocol may permit LP_i to advance its clock during these ticks, but will then recognize a temporal error upon receipt of the message with time-stamp v, and roll back. A rollback at LP_i can itself cause other rollbacks on other LPs, as false messages sent by that LP are undone.

Our goal is not to propose a model that precisely describes all self-initiating parallel simulations, nor is it to analyze the most general possible class of simulation models. Self-initiating models are by no means the most common kind of simulations, and many simulations will not have the power of lookahead that we analyze. However, the analytic modeling of parallel simulations is an art in its infancy. We are simply trying to shed some light on a tractable style of analysis that produces reasonable (and intuitive) results. Even so, despite the many preceding qualifications the proposed model bears a close resemblance to simulations of practical interest. In particular the model accurately describes the behavior of the Ising spin and multiprocessor cache simulations described earlier.

3 Optimal Performance

Finding non-trivial upper and lower bounds on the performance one can achieve in a parallel simulation is an open question. We derive an upper bound on the performance any protocol can achieve under our model assumptions, and derive lower bounds on the performance of a new synchronous conservative protocol. Our bound on the performance of optimistic protocols is independent of the message-cancellation strategy used.

3.1 Upper Bounds

We will present an analytic approach that provides upper bounds on optimal performance for a whole family of lookahead capabilities. Consider any cycle on any LP, and assume that an oracle schedules the processing of that cycle on the earliest possible tick such that no further messages will be received by that LP with a time-stamp less than the time at the end of the cycle. Under our assumptions, this scheduling policy is obviously optimal. Our upper bounds assume the use of this oracle.

Different simulation models have different lookahead abilities. Some models have no lookahead, others are able to predict ahead one cycle, some may be able to predict multiple cycles into the future. For example, the ability of the multiprocessor cache simulation to predict its own future references to global memory is limited only by the memory required to store its trace. We will categorize these abilities by the number of cycles that can be predicted. A simulation model will be said to have J-cycle full-lookahead if the time and content of output messages associated with the completion of cycle k can be predicted at the completion of cycle $(k-J)$. We assume that simulations exploiting J-cycle full-lookahead will always "pre-send" a message J cycles before the message's associated cycle.

The basic approach to constructing an upper bound is simple. The *Global Virtual Time* [7] at tick i is denoted $GVT(i)$; this quantity is the least clock value among all LPs at tick i, and is typically used to gauge the progress of the simulation. We desire to bound the limiting rate of simulation time increase, $\lim_{i\rightarrow\infty} GVT(i)/i$. Our approach is to bound $GVT(i)$ by a function $N(i)$, which is the minimum time-stamp among the "next" messages sent by LPs who received the minimum-time stamped message at tick $i-1$. We then appeal to asymptotic arguments to estimate $\lim_{i\rightarrow\infty} N(i)/i$, and hence bound the limiting rate of simulation time increase.

The bound is constructed as follows. Let $t_{\min}(i)$ be the least time-stamp among all messages sent at tick i. This value need not be equal to $GVT(i)$, because the LP with least clock may not have sent a message, being prevented from doing so by the knowledge that an impending message will arrive with time-stamp less than its next cycle time. Let $r(i,j)$ be the index of the jth LP (among K) who receives the message with time-stamp $t_{\min}(i)$ at tick i. Consider any $LP_{r(i,j)}$, and suppose time $t_{\min}(i)$ falls within the simulation time span encompassed by its $n_j th$ cycle. The next message $LP_{r(i,j)}$ sends cannot have a time-stamp larger than $C_{r(i,j)}(n_j+J)$, the time associated with the end of its $(n_j+J)th$ cycle. The gap of simulation time between $t_{\min}(i)$ and $C_{r(i,j)}(n_j+J)$ is composed of the sum of a number of random variables: a cycle *residual* $C_{r(i,j)}(n_j)-t_{\min}(i)$, plus J cycle time random variables (a J-fold convolution of $\mathcal{F}$). This is illustrated by Figure 1

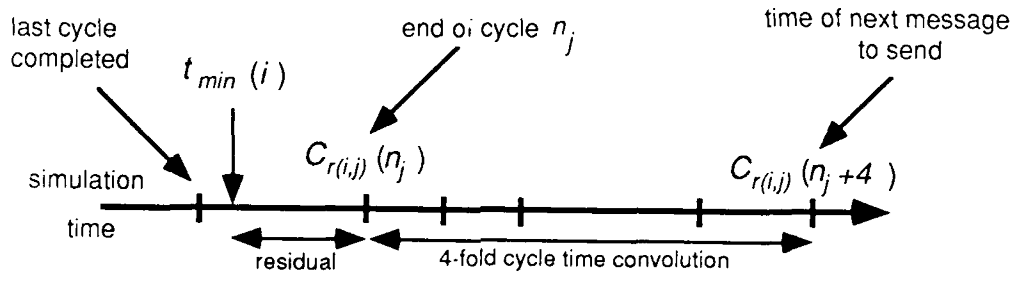

Figure 1: Cycle residual and lookahead cycles for $LP_{r(i,j)}$ in model with $J = 4$ cycle full-lookahead

$t_{\min}(i+1)$ cannot be larger than the least time-stamp on any message sent at the end of tick $i+1$ by one of $LP_{r(i,1)}, \ldots, LP_{r(i,K)}$. Call this latter time-stamp $N(i+1)$. Observe that we may write

$$N(i+1) = t_{\min}(i) + M_{K,J}(i+1),$$

where

$$M_{K,J}(i+1) = \min_{1 \le j \le K} \{C_{r(i,j)}(n_j) - t_{\min}(i) + \mathcal{F}_j(J)\},$$

$\mathcal{F}_j(J)$ being a J-fold convolution of $\mathcal{F}$ random variables. Finally, observe that $N(i+1) \ge GVT(i+1)$. Since our object is to bound $\lim_{i\to\infty} GVT(i)/i$, it will suffice to bound $\lim_{i\to\infty} N(i)/i$.

Observe that for all $i > 0$,

$$\begin{aligned}
N(i)/i &= \sum_{j=1}^{i} (N(j) - N(j-1))/i \\
&= \sum_{j=1}^{i} (t_{\min}(j-1) + M_{K,J}(j) - N(j-1))/i \\
&\le \sum_{j=1}^{i} (N(j-1) + M_{K,J}(j) - N(j-1))/i \\
&= \sum_{j=1}^{i} M_{K,J}(j)/i \qquad (1)
\end{aligned}$$

The appendix gives heuristic reasons for expecting that if $\mathcal{F}$'s tails aren't too large, (e.g., if $\mathcal{F}$ is NBUE, or has an increasing hazard rate function), then it is reasonable to assume that the sequence $\{M_{K,J}(i)\}$ converges into a wide-sense stationary process having finite correlation time [8]. While the supposition is technical, for our purposes here it implies that the limiting value of sum (1) converges to $\Psi(K, J) = \lim_{i\rightarrow\infty} E[M_{K,J}(i)]$. $\Psi(K, J)$ then bounds the limiting rate of increase in GVT.

The preceding discussion leads to our first proposition.

Proposition 1 *For every tick i let $t_{\min}(i)$ be the least time-stamp among all messages sent at the end of tick i, and let $LP_{r(i,1)}, \ldots, LP_{r(i,K)}$ be the set of LP who receive the $t_{\min}(i)$-time message. Let n_j be the cycle index of the $LP_{r(i,j)}$ cycle containing time $t_{\min}(i)$, and $\mathcal{F}_j(J)$ be a convolution of J random variables having distribution $\mathcal{F}$. Define*

$$M_{K,J}(i+1) = \min_{1\leq i\leq K}\{C_{r(i,j)}(n_j) - t_{\min}(i) + \mathcal{F}_j(J)\},$$

and let $\Psi(K, J) = \lim_{i\rightarrow\infty} E[M_{K,J}(i)]$. If the sequence $M_{K,J}(1), M_{K,J}(2), \ldots$ converges to a wide-sense stationary process with finite correlation time, then

$$\lim_{i\rightarrow\infty} GVT(i)/i \leq \Psi(K, J).$$

■

A second proposition follows from the observation that a simulation advancing time at rate $q\mu$ has an average processor utilization of $q\%$.

Proposition 2 *Let the conditions of Proposition 1 be satisfied, and let μ be the mean of $\mathcal{F}$. Then the average processor utilization is no greater than $\Psi(K, J)/\mu$.*

■

We must estimate $\Psi(K, J)$ before these propositions yield any insight on performance. Reconsider the definition of $M_{K,J}(i)$. One takes the minimum of K random variables; each random variable includes the excess time after $t_{\min}(i)$ of the cycle containing $t_{\min}(i)$. We call this time difference a residual. A similar concept is studied in renewal theory, the residual life of a renewal processes. The difference between our residuals and those of renewal theory is that our $t_{\min}(i)$ is itself variable, whereas renewal theory considers the residual following a constant time t. However, in the Appendix we show that if $\mathcal{F}$ is non-lattice and $t_{\min}(i)$ is independent of $LP_{r(i,j)}$, then the limiting residual life has the same distribution as that derived in renewal theory. This limiting distribution is the *equilibrium distribution*[24] of $\mathcal{F}$, called $\mathcal{F}_e$. It is not completely unreasonable to to assume for the purposes of approximation that $t_{\min}(i)$ is independent of all the $LP_{r(i,j)}$, due to the fact that the set of recipients of the $t_{\min}(i)$-time message were chosen uniformly at random from the entire collection of LPs. To the extent that this is a reasonable approximation, as k grows, $M_{K,J}(k)$ increasingly becomes the minimum of K independent and identically distributed (iid) random variables, each the sum of an $\mathcal{F}_e$ random variable and an independent J-fold convolution of $\mathcal{F}$. Alternately, if $\mathcal{F}$ is geometric then its residual life has the same geometric distribution, due to the memoryless property.

Our assumption of random communication patterns is now needed again. When the number of LPs is large compared to K, and when the partners of each communication are chosen randomly, we may take the K random variables comprising $M_{K,J}(k)$ as being independent. This is not rigorously true, as there is a very slight dependence of the residuals on the fact that their LP's did not send the $t_{\min}$-time message; since this is true of all but one LP in the entire system, the assumption of independence is reasonable. Our approximation of $\Psi(K,J)$ is denoted $\Lambda(K,J)$, and is given by

$$\begin{aligned}\Psi(K,J) &\approx E[\min\{K \text{ iid } \mathcal{F}_e + \mathcal{F}(J) \text{ random variables}\}] \\ &\equiv \Lambda(K,J).\end{aligned} \tag{2}$$

Throughout this paper we will implicitly assume the validity of this approximation as a hypothesis to each proposition. The form of the approximation is especially nice, as it permits us to analyze some special cases.

3.1.1 NBUE Distributions

The case of $J = 0$ is of special interest, as it concerns simulations with no lookahead ability. Furthermore, consider simulations where $\mathcal{F}$ is non-lattice and *New Better Than Used in Expectation*, (NBUE) [24][4]. Many common distributions are NBUE, including normals truncated to be positive, gammas, Weibulls, and sums of nonnegative constants with exponentials. When $\mathcal{F}$ is NBUE, then $\mathcal{F}_e$ is dominated stochastically[5] by the exponential with mean μ. Thus, if we replace each $\mathcal{F}_e$ random variable in the definition of $\Lambda(K,0)$ with an exponential having mean μ, the resulting mean μ/K is at least as large as $\Lambda(K,J)$.

Proposition 3 *Suppose that $\mathcal{F}$ is non-lattice and NBUE. Then $\Lambda(K,0) \leq \mu/K$. The optimal processor utilization in a no-lookahead simulation where $\mathcal{F}$ is NBUE is no greater than $1/K$.*

■

This result shows the strong influence that K has on performance when $\mathcal{F}$ is non-lattice—it limits processor utilization to $1/K$. If K remains proportional to N as N increases we have the following result.

[4] A non-negative random variable X is NBUE if for all $t \geq 0$, $E[X|X \geq t] \leq E[X]$. In other words, the expected residual life of X is never greater than the expected value of X.

[5] X is said to dominate Y stochastically if $\Pr\{X > t\} \geq \Pr\{Y > t\}$ for all t.

Proposition 4 *Let $\mathcal{F}$ be non-lattice and NBUE, and suppose $K \geq 3N$ for all N in a no-lookahead simulation. Then under any protocol and for any N, the average number of total cycles processed by the system per tick cannot exceed $1/3$.*

■

K equals N in the cache simulation we have already described, because one LP's global reference is sent to all other LPs. The conclusions of Proposition 4 apply whenever a synchronization protocol treats the model as having no lookahead.

3.1.2 Geometric Distribution

Propositions 3 and 4 depend on $\mathcal{F}$ being non-lattice. When $\mathcal{F}$ is geometrically distributed with mean $\mu = 1/p$, then the residuals defining $\Lambda(K, 0)$ are also geometric with mean $1/p$. It is straightforward to compute $\Lambda(K, 0)$ as the expected minimum of K independent geometrics. This minimum has the same distribution as one geometric with mean $1/(1-(1-p)^K)$. Observe that this mean is always at least one, it cannot diminish arbitrarily as K is increased. When either p is small or K is large the mean is very close to one, leading us to the next proposition.

Proposition 5 *Suppose that $\mathcal{F}$ is geometric with mean $\mu = 1/p$. Then*

$$\Lambda(K, 0) \leq \frac{1}{1-(1-p)^K},$$

and processor utilization is no greater than $p/(1-(1-p)^K)$. Thus, as $(1-p)^K \to 0$, processor utilization is no greater than p.

■

3.1.3 Constant plus Exponential Distributions

Larger upper bounds on utilization are possible given better lookahead ability. Suppose that a simulation has one-cycle full-lookahead ability, and consider the family of distributions where a constant δ is added to an exponential with mean μ_x. Within this family we can decrease the variability by increasing δ, but still retain the tractability of the exponential. This family of random variables is NBUE. In the Appendix we show that under these assumptions

$$\Lambda(K, 1) \leq \delta + \sqrt{\frac{\pi(\delta\mu_x + \mu_x^2)}{2K}} + \frac{2(\delta + \mu_x)\exp\{\frac{-K\delta}{4(\delta+\mu_x)}\}}{K}. \tag{3}$$

The interesting thing to note about this bound is that it decreases in $1/\sqrt{K}$ rather than in $1/K$, as in the no-lookhead case. This suggests that really significant performance gains may be possible when K is moderately large, by exploiting one-cycle full-lookahead. However, the fact that we have increased an upper bound on performance does not necessarily imply that performance itself must increase. In the following section we address this issue by deriving a lower bound on optimal performance under the assumption of full lookahead for $J \geq 1$ cycles.

3.2 Lower Bounds

We now derive a lower bound on optimal performance for simulation models with full-lookaheads of $J \geq 1$ cycles. Our approach is to view synchronization as a scheduling problem, and derive the performance one achieves using a particular scheduling strategy. Being sub-optimal, this performance provides a lower bound on optimal performance. The strategy we study forms the basis for a conservative synchronization protocol.

Consider a simulation model with J-cycle full-lookahead. Recall that we exploit J-cycle full-lookahead by requiring an LP who completes cycle m to predict and send the message associated with the end of cycle $m + J$. Suppose LP_i last executed cycle m, and knows (through some as yet unspecified means) that it will not receive any further messages with a time-stamp t or smaller, t falling within its $(m+k)th$ cycle. LP_i may safely compute cycles $m+1$ through $m+k-1$, and in doing so predict the messages associated with the completion of its cycles $m+1+J$ through $m+k+J-1$. The idea behind our scheduling strategy is to define a *window* by defining this t. All LPs may then advance as described above, whereupon we define a new t and hence a new window. Our strategy defines and processes each window with the following steps.

1. For each LP determine the time of the next message it will send. If LP_i last evaluated cycle m, the time of the next message it will send is $C_i(m+J+1)$. Compute the minimum such $c_{\min}$ among all LPs. $c_{\min}$ is called the *ceiling*.

2. Each LP computes all its cycles with termination times strictly less than $c_{\min}$. For each cycle n that is so processed, the LP predicts and sends the message associated with the completion of cycle $n+J$.

3. Each LP accepts the messages sent in the previous step.

This process is repeated until the simulation termination condition is reached.

The *performance* of this mechanism is derived as follows. Let $c_{\min}(j)$ be the ceiling computed during the jth window. The asymptotic rate at which simulation time advances is identical to the asymptotic rate at which $c_{\min}(j)$ advances. Consider the jth window: every LP_i computes all as-yet-unprocessed cycles with completion times less than $c_{\min}(j)$. Let m_i be the last cycle so processed by LP_i. Note that by the end of the window, LP_i will have sent messages associated with cycles up to $m_i + J$. The time of the next message LP_i will send can be expressed as the sum of $c_{\min}(j)$, the residual of the cycle containing $c_{\min}(j)$, and the cycle time increments of the following J cycles. Therefore, the difference between the time of LP_i's next message and $c_{\min}(j)$ is composed of a cycle residual plus a J-fold convolution of cycle time increments. We have already seen in §3.1 that as j grows large this difference may be approximated as a random variable having the distribution of $\mathcal{F}_e + \mathcal{F}(J)$. Then $c_{\min}(j+1)$ can be expressed as $c_{\min}(j)$ plus the minimum of N such random variables. This shows that

$$E[c_{\min}(j+1) - c_{\min}(j)] \rightarrow \Psi(N, J) \quad \text{as } j \rightarrow \infty.$$

No LP can process more than J cycles in a window, because it can never advance beyond the time of the next message it will send (J cycles distant), computed in step 1 of the window processing. $\Psi(N, J)/J$ consequently bounds the limiting rate at which simulation time advances from below.

Proposition 6 *If the suppositions of Proposition 1 are met, then the limiting rate of simulation time advance using lookahead scheduling on a J-cycle full-lookahead simulation model is at least* $\Psi(N, J)/J$. *The limiting processor utilization is at least* $\Psi(N, J)/(J\mu)$.

■

Now consider the special case of $J = 1$, and the $\delta + \exp\{\mu_x\}$ distribution. In the Appendix we show that

$$\delta + \mu_x\sqrt{\frac{\pi}{2N}} \leq \Lambda(N, 1), \qquad (4)$$

implying that ρ, the average processor utilization achieved, is at least

$$\begin{aligned} \rho &\geq \frac{\delta + \mu_x\sqrt{\frac{\pi}{2N}}}{\delta + \mu_x} \\ &= \frac{r + \sqrt{\frac{\pi}{2N}}}{r + 1} \qquad \text{where } r = \delta/\mu_x. \end{aligned}$$

This shows that the extreme conservatism of having every LP block on the $c_{\min}$-time messages can be highly tempered. If $r = \delta/\mu_x$ is at all significant the utilizations are quite good. For example, if $r = 0.25$ we still get at least 20% utilization. Increase r to 1 and we are assured of 50% utilization. $r = 10$ delivers 91% utilization.

The case where $\mathcal{F}$ is geometric is also of interest. $\Lambda(N, J)$ is composed of the minimum of N random variables, each the sum of J geometric random variables. Each geometric is at least as large as one, implying that $\Lambda(N, J) \geq J$. If the geometrics have mean $\mu = 1/p$, Proposition 6 shows that average processor utilization is at least $100 * p\%$.

4 Analysis of Optimistic Protocol

We next turn to a similar analysis of optimistic protocols. These protocols are complex, especially with regard to the effects of cascading rollbacks. It appears to be a formidable task to put a *lower* bound on the rate at which an optimistic protocol advances simulation time. However, it is easy to extend the ideas of the previous section to put an *upper* bound on this rate. Indeed, our ideas of focusing the analysis on the costs along the critical path are mirrored in Lipton and Mizell's proof that conservative methods cannot arbitrarily outperform Time Warp.

Successful conservative schemes exploit simulation model characteristics such as non-preemptive queueing, and precalculation of event duration times. One of the great hopes for optimistic protocols is that they can be implemented without using explicit knowledge about the simulation model. Consequently, a "model-independent" implementation cannot assume any lookahead. The results of the previous section show that this assumption immediately limits the processor utilizations that are possible. Naturally, the cost of state-saving and rollback limits utilizations even more. To be sure, simulations using Time Warp may exploit model information; indeed, users who have and use Time Warp implementations have suggested they should do so [2]. Note however that these types of optimizations do not alleviate the burden of state-saving. The arguments to follow assume that Time Warp treats the simulation model as though it has no lookahead.

Under Time Warp an LP rolls back if it receives a message with a time-stamp less than its clock. We assume that an LP's state is always saved prior to the execution of a cycle, and model that cost with ticks of length $C_S \geq 1$, as compared to the earlier ticks of length 1. We suppose that a rollback requires C_R time, measured in units where processing an event (without state-saving) takes unit time. The model bounds the rate of simulation time increase on the critical path. The only assumption used by the analysis is that a "late" message causes a rollback at the receiving processor, any effects due to rollback propagation are ignored. A rolled-back processor will re-execute cycles it was rolled past. For this reason our analysis is independent of whether "aggressive" or "lazy" message cancellation[23] is used. By assuming that cycles passed by the rollback are reevaluated, the analysis assumes that "jump forward" mechanisms [6] are **not** used. Under an optimistic protocol a given cycle of an LP may be processed a number of times before it is "cast". To avoid complications we assume that the length in simulation time of a cycle is the same, every time the cycle is processed.

Suppose that a rollback is initiated at LP_i at tick k. Barring any further interruptions due to cascading anti-messages, the rollback completes at tick $k + C_R$. At tick $k + C_R + 1$ the LP rejoins the simulation and communicates its K messages. The idea behind the analysis of optimal performance in §3 was to bound the advance in simulation time between two ticks by looking at the LPs which receive the message with least time-stamp at a tick. Exactly the same idea applies here, except that the increase in simulation time must be measured over more than one tick.

Consider the set of LPs who receive the message with time-stamp $t_{\min}(i)$. Let LP_{min} be the LP in this set with the minimum cycle residual (measured from $t_{\min}(i)$), and recall that $N(i)$ is the tick at which $LP_{\min}$ sends its next message. We will use these definitions to partition the running time into *phases* that measure the time between an LP's receipt of a $t_{\min}$-time message, and the next tick at which the LP completes a cycle and sends a message. The idea of a phase is to measure the rollback delay caused by receipt of the $t_{\min}$-time message. Phase 1 encompasses ticks 1 through $N(1) - 1$. Phase 2 encompasses ticks $N(1)$ through $N(N(1)) - 1$, phase 3 encompasses ticks $N(N(1))$ through $N(N(N(1))) - 1$, and so on. When $\mathcal{F}$ is non-lattice, any LP receiving the $t_{\min}(i)$-time message at the end of tick i must roll back, or already be rolling back. If it is already rolling back and if the transmission of the $t_{\min}$-time message to that LP is independent of the fact it is already rolling back, then on average the rollback is half-way completed. In this case the mean number of ticks in a phase must be at least $1 + C_R/2$. This argument requires $\mathcal{F}$ to be non-lattice, for when $\mathcal{F}$ is discrete it is possible for an LP receiving the least time-stamp message to have the same clock value as the time-stamp, possibly making a rollback unnecessary.

Let $P(i)$ be the number of phases that have completed by tick i, c_i be the tick that completes the phase containing i, and let t_j be the tick that completes the jth phase. Then

$$\begin{aligned}
GVT(i)/i \le GVT(c_i)/i &\le N(c_i)/i \\
&\le \left(\sum_{j=1}^{P(i)+1} M_{K,J}(j) \right) /i \\
&= \left(\frac{\sum_{j=1}^{P(i)+1} M_{K,J}(j)}{P(i)+1} \right) \left(\frac{P(i)+1}{i} \right) \\
&\le \left(\frac{\sum_{j=1}^{P(i)+1} M_{K,J}(j)}{P(i)+1} \right) \left(\frac{P(i)+1}{\sum_{j=1}^{P(i)} (t_j - t_{j-1})} \right).
\end{aligned}$$

We have already identified conditions under which the leftmost quotient converges to $\Lambda(K,0)$. Under similar conditions on the length of phases the rightmost quotient will converge to the reciprocal of the mean number of ticks per phase. Recall that each tick is a factor of C_S slower due to the cost of state-saving. This argument proves our next proposition.

Proposition 7 *Suppose the sequences $\{M_{K,J}(i)\}$ and $\{t_{i+1} - t_i\}$ converge to respective wide-sense stationary processes with finite correlation time. Let $C_{Rh} = C_R/2$. If Time Warp treats a simulation model as though it has no lookahead then*

$$\lim_{i \to \infty} GVT(i)/i \le \begin{cases} \frac{\Psi(K,0)}{C_S(C_{Rh}+1)} & \text{if } \mathcal{F} \text{ is non-lattice} \\ \frac{\Psi(K,0)}{C_S} & \text{if } \mathcal{F} \text{ is discrete} \end{cases}$$

Consequently, the processor utilization is no greater than $\frac{\Psi(K,0)}{\mu C_S(C_{Rh}+1)}$ when $\mathcal{F}$ is non-lattice, and is no greater than $\frac{\Psi(K,0)}{\mu C_S}$ when $\mathcal{F}$ is discrete.

■

An easy upper bound can be put on Time Warp even when it exploits lookahead, because every LP incessantly saves state. Each tick some LP executes a cycle, and first saves state, so that every processing tick is delayed by a state-save. Therefore, Time Warp's performance cannot be any better than a factor of C_S worse than optimal.

Proposition 8 *The optimal rate of simulation time advance under Time Warp on a simulation model having J-cycle full-lookahead is no greater than $\frac{\Psi(K,J)}{JC_S}$; average processor utilization is no greater than $\frac{\Psi(K,J)}{J\mu C_S}$.* ■

5 A Conservative Protocol

The promise of (sometimes) good performance achieved by the scheduling strategy described in §3.2 suggests its use as the basis for a synchronization protocol. Unlike many conservative protocols, this one is synchronous, in that the computation of the ceiling value implicitly contains a global synchronization among processors. This synchronization is all that is needed to implement the policy. The lower bound on performance we derived in §3.2 must change to accommodate the cost of computing the ceiling. Define $C_G \geq 1$ so that a processor is engaged in synchronization overhead $100(1 - 1/C_G)\%$ of the time. Equivalently, one can view the ticks as being C_G percent of the length of our earlier ticks, due to the overhead of synchronization. Depending on the granularity of the event computation the delay cost of synchronization can be quite small, as most richly connected architectures such as a hypercube can compute a global minimum in $\log N$ steps. Some architectures such as the second generation Connection Machine already have hardware support for common global reductions like the minimum. Including this synchronization cost, the lower bound on processor utilizations becomes $\Psi(N,J)/(C_G J\mu)$.

Consider again a simulation with 1-cycle full-lookahead with $\delta + \exp\{\mu_x\}$ time increments. Using approximation (3) and inequalities (3) and (4) we can put a lower bound on the ratio of the conservative protocol's utilization to optimal utilization. Table 1 plots this bound as a function of δ and K, for fixed $\mu_x = 1$, $C_G = 2$ and $N = 65536$.

Relatively good performance is possible when δ is non-trivial relative to μ_x and/or when K is large, even though synchronization overheads are 50%. However, if the upper bound on optimal performance is at all tight there is clearly room for significant improvement. It is here that the extreme conservativeness of having every LP wait for the least-time future message hurts. It may be that more complex protocols such as the bounded-lag protocol [13] could significantly boost performance in this region.

We can determine situations where this conservative protocol achieves better performance than Time Warp. Assume the validity of approximation (3), and assume that $\mathcal{F}$ has the $\delta + \exp\{\mu_x\}$ distribution

$K\backslash\delta$	0.00	0.10	0.5	1.00	5.0	10.0
2	0.001	0.021	0.076	0.116	0.208	0.237
4	0.002	0.034	0.119	0.176	0.295	0.330
8	0.003	0.053	0.173	0.243	0.367	0.399
16	0.004	0.079	0.231	0.303	0.409	0.435
32	0.007	0.114	0.284	0.348	0.434	0.453

(Conservative utilization)/(Optimal utilization)

Table 1: Approximated lower bound on fraction of optimal performance achieved by the conservative protocol when $\mu_r = 1$, $C_G = 2$ and $N = 65536$.

This distribution is NBUE, whence by Propositions 3 and 7, $1/(C_S(C_{Rh}+1)K)$ is an upper bound on Time Warp's utilization. From this, Proposition 6, and equation (4) we determine that the conservative protocol achieves better performance than Time Warp whenever

$$\frac{C_G}{C_S(C_{Rh}+1)} \le \frac{K(\delta + \mu_r\sqrt{\frac{\pi}{2n}})}{\delta + \mu_r}.$$

Note that this inequality assumes that Time Warp is not exploiting lookahead.

Estimates for individual process state sizes in the near term at the Jet Propulsion Lab are from 4K up to 1M bytes [1]. For 4K state sizes, it is estimated that 90% of a processor's time could be devoted to saving state. Using Time Warp on these production problems without the benefit of hardware accelerators, $C_S = 10$ is apparently a reasonable value.

Physical processes modeled by LPs very rarely have zero duration times. Many modeled processes exhibit a fixed startup cost, e.g. chocking a bit into a drill in a manufacturing simulation. Therefore, non-zero values of δ seem reasonable in practice. Relatively large values K are also common, especially in domain oriented simulations where domain sectors are LPs that communicate in a nearest neighbor pattern.

Table 2 plots the ratio of the lower bound on the conservative method's utilization to the upper bound on Time Warp utilization, as a function of δ and K for fixed $\mu_r = 1$, $N = 65536$, $C_G = 2$, and $C_{Rh} = 0$. One set of data assumes that $C_S = 10$. Another assumes that $C_S = 2$, making state-saving comparable to the cost of a global synchronization.

The performance difference is not so great when $\mathcal{F}$ is geometric. Using Propositions 5 and 7 we bound Time Warp's utilization from above by $p/(C_S(1-(1-p)^K)$. A simple lower bound on the utilization of the conservative protocol is p/C_G. Table 3 plots the ratio of these bounds for the same set of parameter values as did Table 2. The values of p are chosen to yield the same mean values of $\mathcal{F}$ as those in Table 2

$K \backslash \delta$	0.00	0.10	0.50	1.00	5.0	10.0
2	0.049	0.954	3.366	5.024	8.341	9.095
4	0.098	1.907	6.732	10.049	16.683	18.191
8	0.196	3.814	13.464	20.098	33.366	36.381
16	0.392	7.629	26.928	40.196	66.732	72.763
32	0.783	15.258	53.856	80.392	133.464	145.526

(Conservative utilization)/(Time Warp utilization)

$\delta + \exp\{\mu_r\}$ distribution, high state-saving costs

$K \backslash \delta$	0.00	0.10	0.50	1.00	5.0	10.0
2	0.010	0.191	0.673	1.005	1.668	1.819
4	0.020	0.381	1.346	2.010	3.337	3.638
8	0.039	0.763	2.693	4.020	6.673	7.276
16	0.078	1.526	5.386	8.039	13.346	14.553
32	0.157	3.052	10.771	16.078	26.693	29.105

(Conservative utilization)/(Time Warp utilization)

$\delta + \exp\{\mu_r\}$ distribution, low state-saving costs

Table 2: Comparison of conservative protocol and Time Warp when $\mathcal{F}$ has the $\delta + \exp\{\mu_r\}$ distribution. $\mu_r = 1$, $N = 65536$, $C_{Rh} = 0$. High state-saving costs modeled with $C_S = 10$, low state-saving costs with $C_S = 2$.

Despite the better showing by Time Warp, in most of the cases shown the conservative method compares favorably with Time Warp. The insensitivity of the conservative method to fanout is a direct consequence of its implicit assumption assumption that the next message to an LP can come from anywhere. This is equivalent to assuming a fanout of N.

Simulation studies suggest that our upper bound on Time Warp's performance is somewhat larger than the observed performance. Figure 2 illustrates the point by plotting the measured (simulated) performance of Time Warp and our conservative method on the analytic model. Comparable overheads are used ($C_S = C_G = 2$), the conservative method exploits a 1-cycle full-lookahead model while Time Warp does not. Aggressive cancellation is used in the Time Warp simulation.

$K\backslash p$	1/1	1/1.1	1/1.5	1/2	1/6	1/11
2	5.000	4.959	4.444	3.750	1.528	0.868
4	5.000	5.000	4.938	4.688	2.589	1.585
8	5.000	5.000	4.999	4.980	3.837	2.667
16	5.000	5.000	5.000	5.000	4.730	3.912
32	5.000	5.000	5.000	5.000	4.985	4.763

(Conservative utilization)/(Time Warp utilization)

geometric distribution, high state-saving costs

$K\backslash p$	1/1	1/1.1	1/1.5	1/2	1/6	1/11
2	1.000	0.992	0.889	0.750	0.306	0.174
4	1.000	1.000	0.988	0.938	0.518	0.317
8	1.000	1.000	1.000	0.996	0.767	0.533
16	1.000	1.000	1.000	1.000	0.946	0.782
32	1.000	1.000	1.000	1.000	0.997	0.953

(Conservative utilization)/(Time Warp utilization)

geometric distribution, low state-saving costs

Table 3: Comparison of conservative protocol and Time Warp when $\mathcal{F}$ has geometric distribution. $\mu_{\mathcal{F}} = 1$, $N = 65536$, $C_{Rh} = 0$. High state-saving costs modeled with $C_S = 10$, low state-saving costs with $C_S = 2$.

6 General Application

The conservative protocol described earlier has broader application than just to our simple analytic model. Its principles form the basis of a parallel simulation testbed we have implemented on an Intel iPSC/2 [19]. The key idea to making efficient use of such a coarse-grained machine is aggregating large numbers of LPs for evaluation on each processor. One advantage to aggregation is that a model which suffers very low processor utilizations when each LP has its own processor can achieve good processor utilizations on a coarse grained machine. For example, consider a model with 65536 LPs, which gets 1% utilization on an architecture with 65536 processors. Evaluate that model on a machine with 64 processors, and on average each processor will have more than 10 events to process each synchronization window. Indeed, the results developed in the framework of a more complex stochastic model show that given 1-cycle full-lookahead, performance of our method approaches optimality as the size of the problem is increased relative to the architecture [20]. These

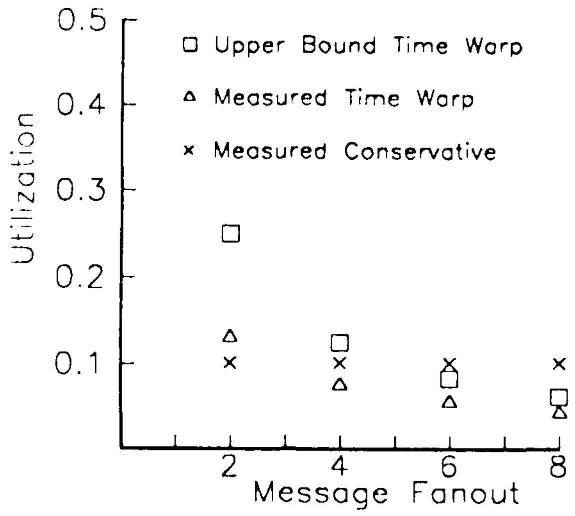

Figure 2: Empirical comparison of *Time Warp* with aggressive cancellation and conservative protocol. Conservative protocol exploits 1-cycle full-lookahead, Time Warp does not. Overheads are equivalent: $C_G = 2$, $C_S = 2$. $N = 65536$, communication patterns are random. $\delta = 0.25$, $\mu_x = 1$.

conclusions are supported by experiments on the testbed where we have achieved processor utilizations in the range of 60% – 85% using 32 processors on large queueing network, logic network, and cellular automaton simulations. The performance degradation there is not due to blocked processors, it is due to communication and synchronization overheads.

The results reported here are easily adapted to simulation models having only time-lookahead. The conservative protocol is modified so that promises of future messages are sent as soon as possible, and then the messages themselves are sent upon completion of the appropriate cycle. The windows are defined as before (the ceiling is the least next appointment time to be sent), but instead of processing all events in one pass, the protocol iterates over the window. Each iteration, computations that are assured of no future messages (as established by the lookahead message times) are performed. These create messages that will "free" other computations that depend on them. The analysis goes through as before, except that the increase in simulation time must be amortized over the average number of iterations in a window. We have not yet firmly established this value, but heuristic arguments suggest that it is $O(\log N)$.

7 Conclusions

This paper proposes and analyzes an intuitive model of massively parallel discrete-event simulations. We derive non-trivial upper and lower bounds on optimal performance for certain classes of simulations, derive an upper bound on Time Warp's performance, and derive a lower bound on the performance of a newly proposed conservative method. These results permit a derivation of sufficient conditions for the conservative method to outperform Time Warp.

Our analysis quantifies the dependence of performance on the time-increment distribution, showing that distributions with significant constant components lead to good performance. We also determine the sensitivity of performance to lookahead, and to message fan-out. Unfortunately, our results rest on approximations which are justified only heuristically, although there is excellent agreement between analytic and empirical results. Future research may be directed towards firming up the foundations of our approach.

Our results are significant in two ways. To our knowledge this is the first analysis able to analytically compare the performance of a synchronization protocol on a stochastic model with a non-trivial bound on the optimal performance one can achieve. It is also significant that we are able to classify simulation models under which a conservative method has *provably* good performance. To be sure, there are a large number of simulation models where our protocol will fail miserably, and there are a large number of models which lack the lookahead demanded by our method. Nevertheless, a better understanding of the complex behavior of parallel simulations demands analysis, and this paper is an early effort at providing that analysis.

Acknowledgements

The opinions of Phil Heidelberger and David Jefferson were instrumental in shaping this paper.

References

[1] C.A. Buzzell, R.M. Fujimoto, and M.J. Robb. Modular VME rollback hardware for time warp. In *Distributed Simulation 1990*, pages 153–156, Society for Computer Simulation, 1990.

[2] D. Baezner et al. Algorithmic optimizations of simulations on time warp. In *Distributed Simulation 1989*, pages 73–78, SCS Simulation Series, 1989.

[3] F. Wieland et al. Distributed combat simulation and time warp: the model and its performance. In *Distributed Simulation 1989*, pages 14–20, SCS Simulation Series, 1989.

[4] Philip Hontalas et al. Performance of the colliding pucks simulation on the time warp operating system. In *Distributed Simulation 1989*, pages 3–7, SCS Simulation Series, 1989.

[5] R. Fujimoto et al. Design and performance of special purpose hardware for time warp. In *15th Annual Int'l Symposium of Computer Architecture*, June 1988.

[6] R. M. Fujimoto. Parallel discrete event simulation. In *Proceedings of the 1989 Winter Simulation Conference*, pages 19–28, Washington, D.C., December 1989. To appear in the Communications of the ACM.

[7] D. R. Jefferson. Virtual time. *ACM Trans. on Programming Languages and Systems*, 7(3):404–425, 1985.

[8] H.J. Larson and B.O. Shubert. *Probabilistic Models in Engineering Sciences*. Volume 1. Wiley, New York, 1979.

[9] S. Lavenberg and R. Muntz. Performance analysis of a rollback method for distributed simulation. In *Performance '83*, pages 117–132. Elsevier Science Pub. B. V. (North Holland), 1983.

[10] Y-B. Lin, J-L. Baer, and E.D. Lazowska. Tailoring a parallel trace-driven simulation technique to specific multiprocessor cache coherence protocols. In *Distributed Simulation 1989*, pages 185–190, SCS Simulation Series, 1989.

[11] Y-B. Lin and E.D. Lazowska. Optimality considerations for "time warp" parallel simulation. In *Distributed Simulation 1990*, pages 29–34, SCS Simulation Series, 1990.

[12] R.J. Lipton and D.W. Mizell. Time warp vs. Chandy-Misra: a worst-case comparison. In *Distributed Simulation 1990*, pages 137–143, SCS Simulation Series, 1990.

[13] B.D. Lubachevsky. Efficient distributed event-driven simulations of multiple-loop networks. *Communications of the ACM*, 32(1):111–123, 1989.

[14] B.D. Lubachevsky. Efficient parallel simulations of asynchronous cellular arrays. *Complex Systems*, 1:1099–1123, 1987.

[15] B.D. Lubachevsky. Scalability of the bounded lag distributed discrete-event simulation. In *Distributed Simulation 1989*, pages 100–107, SCS Simulation Series, 1989.

[16] V. Madasetti, J. Walrand, and D. Messerschmitt. Synchronization in message-passing computers: models, algorithms, and analysis. In *Distributed Simulation 1990*, pages 35–48, SCS Simulation Series, 1990.

[17] B.C. Merifield, S.B. Richardson, and J.B.G. Roberts. Quantitative studies of discrete event simulation modelling of road traffic. In *Distributed Simulation 1990*, pages 188–193, SCS Simulation Series, 1990.

[18] D. Mitra and I. Mitrani. Analysis and optimum performance of two message-passing parallel processors synchronized by rollback. In *Performance'84*, pages 35–50. Elsevier Science Pub. B. V. (North Holland), 1984.

[19] D. Nicol, C. Micheal, and P. Inouye. Efficient aggregaton of multiple LP's in distributed memory parallel simulations. In *Proceedings of the 1989 Winter Simulation Conference*. pages 680-685. Washington. D.C., December 1989.

[20] D.M. Nicol. *The Cost of Conservative Synchronization in Parallel Discrete-Event Simulations*. Technical Report 90-20, ICASE. 1990.

[21] D.M. Nicol. Parallel discrete-event simulation of FCFS stochastic queueing networks. *SIGPLAN Notices*, 23(9):124 137, September 1988.

[22] D. A. Reed. A.D. Maloney, and B.D. McCredie. Parallel discrete event simulation using shared memory. *IEEE Trans. on Software Engineering*, 14(4):541–553, April 1988.

[23] Peter L. Reiher. Richard Fujimoto. Steven Bellenot, and David Jefferson. Cancellation strategies in optimistic execution systems. In *Distributed Simulation 1990*, pages 112–121, Society for Computer Simulation. 1990.

[24] H.S. Ross. *Stochastic Processes*. Wiley, New York, 1983.

[25] D.B. Wagner and E.D. Lazowska. Parallel simulation of queueing networks:limitations and potentials. In *Proceedings of the 1989 SIGMETRICS Conference*, pages 146–155, Berkeley. CA. 1989.

Appendix

In this appendix we derive a number of results too detailed to include in the body of the paper.

Limiting Distribution of Residual

Let $X(t)$ and $Y(t)$ be independent renewal processes having non-lattice inter-renewal distribution F and G respectively. Let $R_X(t)$ denote the ***residual life*** at t of the process $X(t)$—the remaining time until the next renewal. It is known [24] that as t grows large $R_X(t)$ converges in distribution to F's associated *equilibrium distribution* F_e:

$$\Pr\{F_e > s\} = \int_s^\infty \Pr\{F > u\}/\mu \, du$$

where μ is F's mean. Now let S_j be the time of the jth renewal in $Y(t)$. We will sketch an argument showing why the limiting distribution of $R_X(S_j)$ as $j \to \infty$ is also F_e.

To show convergence we must demonstrate that for every $x \geq 0$ and $\epsilon > 0$ there exists a j_ϵ such that for all $j > j_\epsilon$,

$$|\Pr\{R_X(S_j) > x\} - \Pr\{F_e > x\}| \leq \epsilon.$$

Choose any x and ϵ. Let $g_j(t)$ be the density function of S_j (a j-fold convolution of G). By the independence of $X(t)$ and $Y(t)$ we may write

$$\Pr\{R_X(S_j) > x\} = \int_0^\infty \Pr\{R_X(t) > x\} g_j(t) \, dt.$$

and

$$|\Pr\{R_X(S_j) > x\} - \Pr\{F_e > x\}| = \int_0^\infty |\Pr\{R_X(t) > x\} - \Pr\{F_e > x\}| g_j(t) \, dt.$$

Because $R_X(t)$ converges in distribution to F_e as $t \to \infty$, we may choose t_ϵ so large that for all $t > t_\epsilon$, the absolute difference inside the integral above is no greater than $\epsilon/2$. We may also choose some j_ϵ so large that $\Pr\{S_j < t_\epsilon\} < \epsilon/2$ for all $j \geq j_\epsilon$. In this case for all $j \geq j_\epsilon$

$$\begin{aligned} \int_0^\infty |\Pr\{R_X(t) > x\} - \Pr\{F_e > x\}| g_j(t) \, dt &< \int_0^{t_\epsilon} g_j(t) \, dt + \int_{t_\epsilon}^\infty (\epsilon/2) g_j(t) \, dt \\ &\leq \epsilon/2 + \epsilon/2. \end{aligned}$$

This demonstrates that the distribution of $R_X(S_j)$ converges to F_e.

$\lim_{j\to\infty} \sum_{i=1}^{j} M_{K,J}(i)/i \to \Psi(K,J)$

Here we describe reasonable conditions under which the limiting average of the sequence $\{M_{K,J}(i)\}$ converges to $\Psi(K,J)$. View the sequence $M_{K,J}(1), M_{K,J}(2), \ldots,$ as a discrete-time stochastic process $\{M_{K,J}(i)\}$. For i large it is reasonable to assume that this process is stationary in the wide-sense[8], meaning that there exists values $\Psi_{K,J}$ and $C^2_{K,J}$ such that $E[M_{K,J}(i)] = \Psi_{K,J} < \infty$ and $E[M_{K,J}(i)^2] = C^2_{K,J} < \infty$ for all i sufficiently large, and that $Cov[M_{K,J}(i), M_{K,J}(j)]$ is a function only of $|i-j|$. Furthermore, we expect that $M_{K,J}(i)$ and $M_{K,J}(j)$ should become independent as $|j-i|$ grows. If they become independent quickly enough, we will have

$$\sum_{j=i+1}^{\infty} |E[M_{K,J}(i)M_{K,J}(j)] - \Psi^2_{K,J}| < \infty. \qquad (5)$$

If this inequality holds true (for a general wide-sense stationary process) the process is said to have a finite *correlation time*.

We have not developed formal arguments that $\{M_{K,J}(i)\}$ becomes wide-sense stationary with a finite correlation time. However, we can give heuristic reasons why it is reasonable to assume so. $M_{K,J}(i)$ is the minimum of K random variables, each comprised of the sum of a residual plus the sum of J cycle times. Among all these let $n_{\max}$ be the maximum index of a cycle time random variable (or residual) appearing in $M_{K,J}(i)$. Now suppose the time-increment distribution is exponential. Any $M_{K,J}(j)$ composed entirely of random variables from cycles with indices greater than $n_{\max}$ is independent of $M_{K,J}(i)$, owing to the memoryless property of the exponential. Let D be the random number of ticks that pass after i before every LP has evaluated cycle $n_{\max}$. Then we have

$$\sum_{j=i+1}^{\infty} |E[M_{K,J}(i)M_{K,J}(j)] - \Psi(K,J)^2| = \sum_{j=i+1}^{i+D} |E[M_{K,J}(i)M_{K,J}(j)] - \Psi(K,J)^2|.$$

Since $M_{K,J}(i)$ and $M_{K,J}(j)$ have the same distribution, we must have $E[M_{K,J}(i)M_{K,J}(j)] \le C^2_{K,J}$. Thus

$$\sum_{j=i+1}^{i+D} |E[M_{K,J}(i)M_{K,J}(j)] - \Psi(K,J)^2| \le E[D]\,(C^2_{K,J} - \Psi^2_{K,J}) < \infty$$

provided that $E[D]$ is finite. Assuming that a serial simulation will always advance a given finite amount of simulation time in a finite expected number of cycle executions, $E[D]$ will be finite. This is true because a serial simulation will always advance the LP with least next message time each tick, and in a finite expected number of steps will advance each LP at least once. $E[D]$ is no larger than this expectation, and is hence finite. This argument rests on the fact that $M_{K,J}(i)$ and $M_{K,J}(j)$ become independent once j is large enough. The independence is an artifact of the exponentiality. Intuitively, if the tail of $\mathcal{F}$ cannot become too large (e.g. if $\mathcal{F}$ is NBUE or if it has an increasing hazard rate function), it is reasonable to expect rapidly diminishing correlation and hence a finite correlation time. Such technical details appear to be difficult to establish.

The result we want follows if $\{M_{K,J}(i)\}$ is wide-sense stationary with a finite correlation time. Let $\overline{M_{K,J}}(i)$ be the average $M_{K,J}$ value taken over the first i ticks:

$$\overline{M_{K,J}}(i) = \frac{\sum_{j=1}^{i} M_{K,J}(j)}{i}.$$

From [8](p. 484) we find that

$$\lim_{i \to \infty} E[(\overline{M_{K,J}}(i) - \Psi(K,J))^2] = 0.$$

This condition is sufficient strong for us to take $\Psi(K,J)$ as the limiting value of $\overline{M_{K,J}}(i)$.

Expected Minimum of Exponential Sums

Next we derive upper and lower bounds on the expected minimum of n independent and identically random variables, each constructed by adding two independent exponentials which need not have the same mean (this is a slight generalization of an Erlang-2 distribution).

Our approach is to analyze the hazard rate function of a single exponential-sum. We will construct one random variable with a larger hazard rate, and one with a smaller one. The former is stochastically smaller than the exponential-sum, hence the minimum of n such is stochastically smaller than the minimum of n exponential-sums. Similarly, the minimum of n independent random variables that stochastically dominate an exponential sum will stochastically dominate the minimum of n exponential-sums.

Let λ_1, λ_2 with $\lambda_1 \geq \lambda_2$ be the exponential parameters for exponentials X_1 and X_2. The hazard rate function $\lambda_s(t)$ of $X_1 + X_2$ is found by considering a two-stage process where the the first stage requires X_1 time and the second stage requires X_2. *Intuitively* $\lambda_s(t)$ is the instantaneous probability density associated with the process finishing at t, given that it has not yet finished. Condition on whether $X_1 < t$: if so, *then* $\lambda_s(t)$ is λ_2 because the process is in the second stage, if not it is zero because the process has not yet finished the first stage. Thus we have

$$\lambda_s(t) = (1 - \Pr\{X_1 > t | X_1 + X_2 > t)\lambda_2. \qquad (6)$$

An equivalent (but much nastier) expression for λ is derivable from first principles, taking the quotient of the density function of $X_1 + X_2$ to the probability that $X_1 + X_2 > t$. Our expression is more convenient, in that it suggests simple ways in which $\lambda_s(t)$ can be bounded from above and below.

An upper bound on $\lambda_s(t)$ is constructed by observing that the conditional probability in equation (6) is at least as large as $\Pr\{X_1 > t\} = \exp\{-\lambda_1 t\}$. Thus

$$\lambda_s(t) \leq (1 - \exp\{-\lambda_1 t\})\lambda_2.$$

This latter function is concave in t and is hence dominated everywhere by the line tangent to it at $t = 0$: $h_l(t) = t\lambda_1\lambda_2$. A random variable with hazard rate function $h_l(t)$ is therefore stochastically dominated by $X_1 + X_2$.

An lower bound on $\lambda_s(t)$ is constructed by exploiting the fact that

$$\Pr\{X_1 > t | X_1 + X_2 > t\} = \frac{\Pr\{X_1 > t\}}{\Pr\{X_1 + X_2 > t\}}$$

$$\begin{aligned}
&\leq \frac{\Pr\{X_1 > t\}}{\Pr\{X_1 + X_3 > t\}} \quad \text{where } X_3 \text{ has the distribution of } X_1 \\
&= \frac{1}{1+\lambda_1 t}.
\end{aligned}$$

Consequently,

$$\lambda_s(t) \geq \left(1 - \frac{1}{1+\lambda_1 t}\right)\lambda_2.$$

This function is increasing and concave in t. It equals $\lambda_2/2$ when $t = 1/\lambda_1$. Consequently, this function dominates the piecewise linear function $h_u(t)$ which rises linearly with slope $\lambda_1\lambda_2/2$ until $t = 1/\lambda_1$, and then is the constant $\lambda_2/2$.

Let $Z_1, \ldots, Z_n$ be n independent random variables with hazard rate $h_u(t)$. The hazard rate of the minimum of these is simply $n \cdot h_u(t)$. To compute the expected minimum we use a well-known relationship between a random variable's hazard rate function and its cumulative distribution function. We have

$$\begin{aligned}
E[\min\{Z_1,\ldots,Z_n\}] &= \int_0^\infty \Pr\{\min\{Z_1,\ldots,Z_n\} > t\}\, dt \\
&= \int_0^\infty \exp\{-\int_0^t n \cdot h_u(s) ds\}\, dt \\
&= \int_0^{1/\lambda_1} \exp\{-n\lambda_1\lambda_2 t^2/4\}\, dt + \int_{1/\lambda_1}^\infty \exp\{-(tn\lambda_2/2 - n\lambda_2/(4\lambda_1))\}\, dt \\
&= \int_0^{1/\lambda_1} \exp\{-n\lambda_1\lambda_2 t^2/4\}\, dt + \frac{2\exp\{-n\lambda_2/(4\lambda_1)\}}{(n\lambda_2)}.
\end{aligned}$$

To evaluate the remaining integral we make the change of variables $s = t\sqrt{n\lambda_1\lambda_2/2}$, and discover that

$$\begin{aligned}
\int_0^{1/\lambda_1} \exp\{-n\lambda_1\lambda_2 t^2/4\}\, dt &= \sqrt{\frac{2}{n\lambda_1\lambda_2}} \int_0^{\sqrt{n/2}} \exp\{-s^2/2\}\, ds \\
&= \sqrt{\frac{4\pi}{n\lambda_1\lambda_2}} \left(\Theta(\sqrt{n/2}) - 1/2\right),
\end{aligned}$$

where $\Theta()$ is the cumulative distribution function for a standard normal. This gives the upper bound

$$\begin{aligned}
E[\text{minimum of } n \text{ iid exponential-sums}] &\leq \sqrt{\frac{4\pi}{n\lambda_1\lambda_2}}(\Theta(\sqrt{n/2}) - 1/2) + \frac{2\exp\{-n\lambda_2/(4\lambda_1)\}}{(n\lambda_2)} \\
&\leq \sqrt{\frac{\pi}{n\lambda_1\lambda_2}} + \frac{2\exp\{-n\lambda_2/(4\lambda_1)\}}{(n\lambda_2)}. \qquad (7)
\end{aligned}$$

A lower bound is found similarly. The hazard rate for the minimum of n independent random variables having hazard rate function $h_l(t)$ is $n \cdot h_l(t)$. Integrating as we did to derive the upper bound, we determine a lower bound of

$$\sqrt{\frac{\pi}{2n\lambda_1\lambda_2}} \leq E[\text{minimum of } n \text{ iic' exponential-sums}] \qquad (8)$$

Bounds on $\delta + \exp(\mu_x)$ Distribution

Next we consider some bounds on $\Lambda(K, 1)$ derivable when the time-increment distribution is a constant δ plus an exponential with mean μ_x, and when the simulation has 1-cycle full-lookahead.

$\Lambda(K, 1)$ is the expected minimum of K random variables, each comprised of a residual plus a time-increment value. The residual has the distribution of the equilibrium distribution of the $\delta + \exp(\mu_x)$ distribution. We first consider this equilibrium distribution. Working directly from definitions [24], we determine that its hazard rate is

$$h(t) = \begin{cases} \frac{1}{\delta+\mu_x - t} & \text{for } t \leq \delta \\ \frac{1}{\mu_x} & \text{for } t > \delta \end{cases}$$

Since $h(t) \geq 1/(\delta + \mu_x)$ for all t, the equilibrium distribution is stochastically dominated by an exponential with mean $\delta + \mu_x$. Let R_i be a residual having this equilibrium distribution, R_i^x be an exponential with mean $\delta + \mu_x$, and X_i be exponential with mean μ_x. Then the sum $R_i + \delta + X_i$ is stochastically dominated by $R_i^x + \delta + X_i$, and

$$\begin{aligned} E[\min_{1\leq i\leq K}\{R_i + \delta + X_i\}] &\leq \delta + E[\text{minimum of K iid exponential-sums, parameters } 1/(\delta + \mu_x) \text{ and } 1/\mu_x] \\ &\leq \delta + \sqrt{\frac{\pi(\delta\mu_x + \mu_x^2)}{2n}} + \frac{2(\delta + \mu_x)\exp\{\frac{-n\delta}{4(\delta+\mu_x)}\}}{n}. \end{aligned}$$

R_i stochastically dominates an exponential with mean μ_x. Consequently a lower bound on the minimum of interest is found by replacing R_i with such an exponential. Then

$$\begin{aligned} E[\min_{1\leq i\leq K}\{R_i + X_i\} &\geq \delta + E[\text{minimum of K iid exponential-sums, both parameters } 1/\mu_x\ ] \\ &\geq \delta + \mu_x\sqrt{\frac{\pi}{2n}}. \end{aligned}$$

NASA

Report Documentation Page

1. Report No. NASA CR-182010 ICASE Report No. 90-21	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle PERFORMANCE BOUNDS ON PARALLEL SELF-INITIATING DISCRETE-EVENT	5. Report Date March 1990
	6. Performing Organization Code
7. Author(s) David M. Nicol	8. Performing Organization Report No. 90-21
	10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	11. Contract or Grant No. NAS1-18605
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Submitted to ACM Transactions on Modelling and Computer Simulations

Final Report

16. Abstract

This paper considers the use of massively parallel architectures to execute discrete-event simulations of what we term "self-initiating" models. A logical process in a self-initiating model schedules its own state re-evaluation times, independently of any other logical process, and sends its new state to other logical processes following the re-evaluation. Our interest is in the effects of that communication on synchronization. We consider the performance of various synchronization protocols by deriving upper and lower bounds on optimal performance, upper bounds on Time Warp's performance, and lower bounds on the performance of a new conservative protocol. Our analysis of Time Warp includes the overhead costs of state-saving and rollback. The analysis points out sufficient conditions for the conservative protocol to outperform Time Warp. The analysis also quantifies the sensitivity of performance to message fan-out, lookahead ability, and the probability distributions underlying the simulation.

17. Key Words (Suggested by Author(s)) discrete-event simulation, parallel algorithms, performance analysis	18. Distribution Statement 62 - Computer Systems 66 - Systems Analysis Unclassified - Unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 29	22. Price A03

NASA FORM 1626 OCT 86

NASA-Langley, 1990